Lab 2

Introduction:

Wireless Access in Vehicular Environments (WAVE), based on the IEEE 802.11p standard and IEEE 1609 family of standards, uses application-level User Priorities (UPs) to differentiate and prioritize various types of application traffic. There are eight UP levels (0-7), which represent different application-level traffic categories. These priorities help manage data transmission by ensuring that critical information is sent with high priority.

The eight application-level UPs are mapped to four Access Categories (ACs) at the MAC sublayer, which uses EDCA (Enhanced Distributed Channel Access). This mapping is shown in Table 1.

Table 1: UP-to-AC mappings

Priority	UP (Same as 802.1D user priority)	802.1D designation	AC	Designation (informative)
Lowest ↓ Highest	1	BK	AC_BK	Background
	2	—	AC_BK	Background
	0	BE	AC_BE	Best Effort
	3	EE	AC_BE	Best Effort
	4	CL	AC_VI	Video
	5	VI	AC_VI	Video
	6	VO	AC_VO	Voice
	7	NC	AC_VO	Voice

For each channel (SCH or CCH), every station maintains four transmit queues, one for each AC, and four independent EDCAFs (Enhanced Distributed Channel Access Functions), one for each queue. EDCAF is an enhanced version of DCF (Distributed Coordination Function), and it contends for the medium on the same principles of CSMA/CA, but based on the parameters specific to the AC it is contending for. These parameters, referred to as EDCA parameters, are:

I. Arbitration Inter Frame Space (AIFS): The time period the medium is sensed idle before the transmission or backoff is started.

II. CW_{min}, CW_{max}: The minimum and maximum Contention Window size limits used for backoff.

III. Transmission Opportunity (TXOP) limits: The maximum duration of the transmission after the medium is acquired. The TXOP Limit of zero indicates that only one frame can be transmitted in a TXOP.

The values of EDCA parameters are different for different ACs. The higher priority ACs wait a small AIFS time period, while the lower priority ACs have to wait a longer AIFS time before they can access the medium. The size of the Contention Window (CW) varies such that the higher priority ACs choose backoff values from a smaller window compared to the lower priority ACs. TXOP Limit is also set in a way that the higher priority ACs get access to the medium for longer durations (In WAVE, TXOP Limit is set to zero for all ACs). As the values of EDCA parameters are AC-specific, they are sometimes referred to as AIFS[AC], CW_{min}[AC], CW_{max}[AC] and TXOP[AC]. Thus, basically, the main difference between DCF and EDCAF is that EDCAF uses AC-specific parameters AIFS[AC], CWmin[AC], and CWmax[AC] instead of using fixed values DIFS, CW_{min}, and CW_{max}.

Table 2 shows the values of EDCA parameters for each AC. Here are the default values for the parameters mentioned in Table 2 and other related parameters:

- $aCW_{min} = 15 \times aSlot$
- $aCW_{max} = 1023 \times aSlot$
- $aSlot = 13\mu s$
- $AIFS[AC] = AIFSN[AC] \times aSlot + SIFS$
  where $AIFSN$ is Arbitration Inter Frame Space Number
- $SIFS = 32\mu s$

Table 2: —Default values for EDCA parameters

AC	CWmin	CWmax	AIFSN	TXOP Limit OFDM/CCK-OFDM PHY
AC_BK	aCWmin	aCWmax	9	0
AC_BE	aCWmin	aCWmax	6	0
AC_VI	(aCWmin+1)/2–1	aCWmin	3	0
AC_VO	(aCWmin+1)/4–1	(aCWmin+1)/2–1	2	0

The four EDCAFs at the AC transmit queues behave like virtual stations inside the real station such that each EDCAF contends for the medium independently of other EDCAFs. Thus, there exist two levels of contention: internal contention among different EDCAFs/ACs inside the same station, and external contention among different stations. This may result in a situation where more than one EDCAF in the same station counts their backoff timers

to zero and tries to transmit at the same time slot. This leads to a situation referred to as internal collision or virtual collision. In such a situation, the access to the medium (TXOP) is granted to the EDCAF with the highest priority AC among the colliding EDCAFs, and the lower priority colliding EDCAFs double their CWs and choose new backoff values, just as if an external collision occurred. An external collision occurs if backoff timers of EDCAFs at two or more stations reach zero at the same (or close enough) time slot(s).

In LAB 1, only external collisions were considered, where all WSMs sent by vehicles have the same UP (UP=5 , which is mapped to AC_VI). So, no internal collisions were involved. In this LAB, we want to introduce internal collisions to the scenario in LAB 1.

Scenario description:

In this scenario, a Roadside Unit (RSU) communicates with On-Board Units (OBUs) installed in vehicles. Vehicles can enter and exit the RSU's coverage area. We assume that each OBUs and the RSU are equipped with a single radio transceiver. Consequently, a channel-switching mode is necessary to alternate between the Control Channel (CCH) and Service Channels (SCHs). During each Control Channel Interval (CCHI), the RSU periodically broadcasts two WAVE Service Advertisements (WSAs) on the CCH to announce the availability of two specific WAVE services (Service_1 and Service_2) on one of the SCHs (assume both services are advertised to take place on channel CH174). Those two services have different UPs such that they are mapped to different ACs. Thus, internal collisions are possible. When an OBU receives a WSA that advertises a service of interest, it switches to the specified SCH during the Service Channel Interval (SCHI) time slot to engage in the advertised activity. This activity involves participants broadcasting some data (such as streaming services, navigation aids, or carpooling information) with the specified UP to nearby participants and the RSU, as shown in Figure 1. We assume all vehicles that receive the WSAs will join the advertised activities. During each synchronized interval, some participants may leave the activities, while new ones may join.

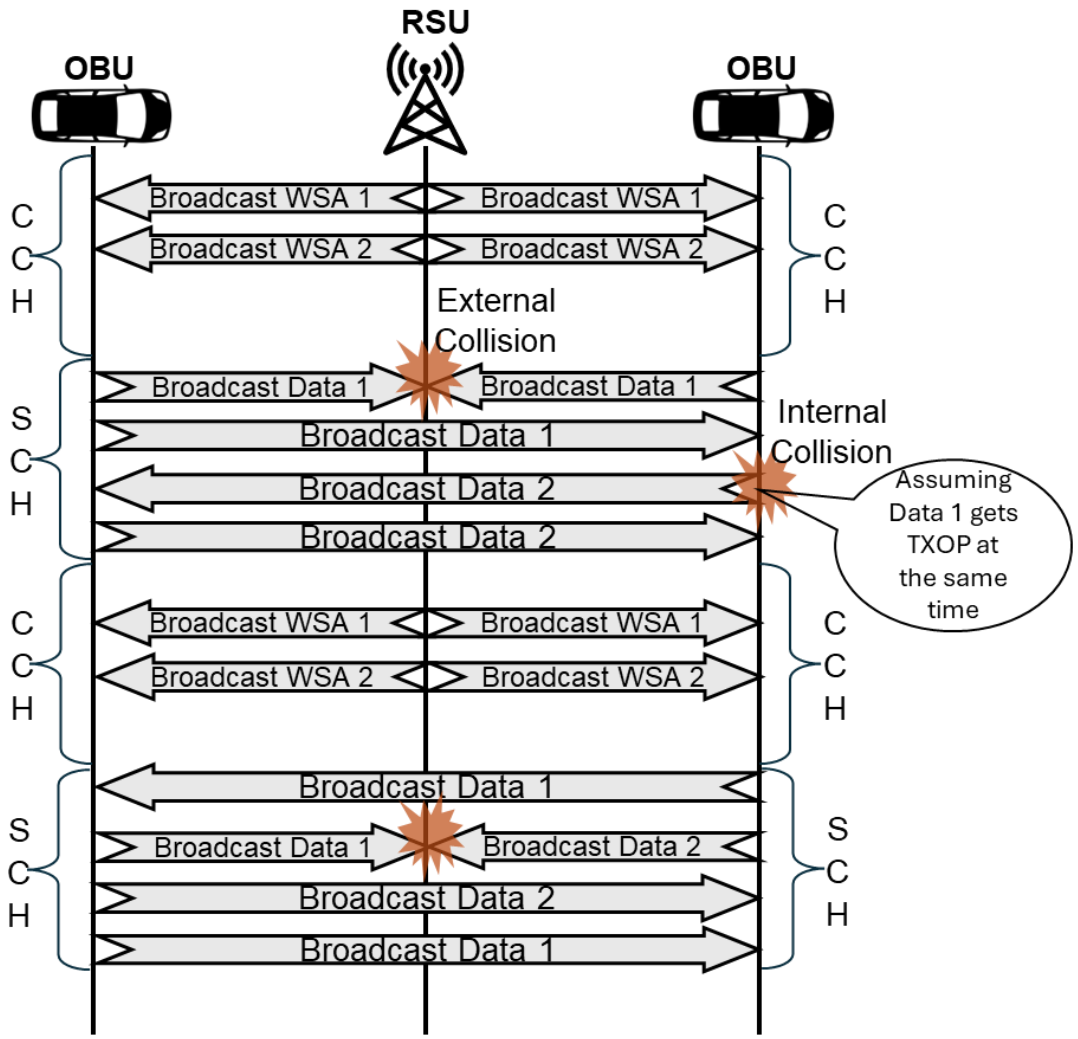

Figure 1: Communication workflow

Simulation Description:

The SUMO traffic scenario is the same as in LAB 1. The scenario features a one-direction, two-lane ring road with a total length of 3040 meters, as depicted in Figure 2. The average speed of vehicles on this road is 13.89 meters per second. Each vehicle travels within the communication range of the RSU for about 760m. When a vehicle exits the RSU's communication range, a new vehicle is randomly placed within the range. Thus, the number of vehicles at a given time is fixed. The RSU is strategically located at the center of the ring road. Assume an ideal wireless channel (i.e. no errors due to noise or other channel impairments).

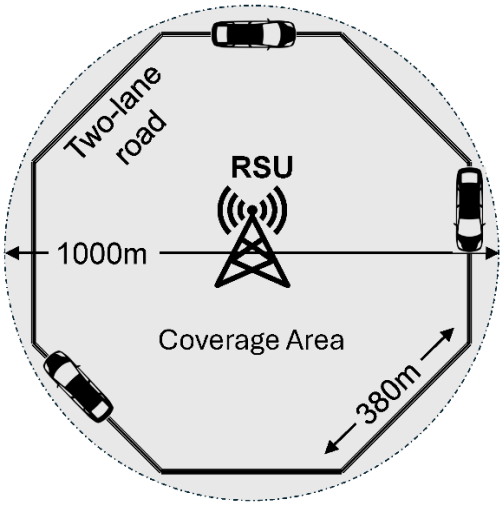

Figure 2: Physical layout of the scenario

At the application level, each vehicle is required to run two applications: App_1 for Service_1 and App_2 for Service_2. However, the Veins framework does not implement the WAVE Short Message Protocol (WSMP), which defines the network and transport layer services in WAVE. In Veins, the application layer is placed directly on top of the MAC layer, bypassing the Network/transport layers. Since the transport layer is responsible for functions like multiplexing and demultiplexing application data using port numbers, the absence of this layer in Veins makes it challenging to manage multiple independent applications running simultaneously on the same vehicle.

To mimic the behavior of running two separate applications (Apps) using a single App instance, the application running on each vehicle needs to generate two distinct WSM (WAVE Short Message) types, each associated with a different User Priority (UP).

To achieve this, you will need to modify several files. In the "**MyVeinsCarApp.ned**" file, introduce two new variables, such as avg_wsmInterval_app2 and wsmInterval_app2, which will define the interarrival time for WSM packets generated by App2. These names are just examples, and you may choose more appropriate names or even rename the existing variables as well to clearly distinguish them for App1. Additionally, you should update the omnetpp.ini file to define avg_wsmInterval_app2 and assign suitable values for App2's packet generation intervals. In the "**MyVeinsCarApp.h**" and "**MyVeinsCarApp.cc**" files, new variables will be required to store App2-specific parameters, which will be responsible for handling the logic for packet generation related to the second application.

In the "**MyVeinsCarApp.cc**" file, you will also need to modify the MyVeinsCarApp::onWSA(DemoServiceAdvertisment* wsa) function to ensure that it starts generating WSMs when the corresponding WSA (WAVE Service Advertisement) for

APP1/App2 is received for the first time. This will allow the application to properly generate and manage two distinct types of WSMs based on the received WSA.

For the RSU, adjustments will be necessary to ensure that it generates two WSAs during each Control Channel Interval (CCHI). You can achieve this by overriding the function DemoBaseApplLayer::startService(Channel channel, int serviceId, string serviceDescription) to schedule and broadcast two different WSAs, each representing a different service. This will make the RSU manage both services simultaneously.

Also, you need to modify the application layer's data collection mechanisms to distinguish between the data generated by App1 and App2. This distinction is critical for the correct handling and analysis of statistics of each App separately.

In addition to the previously mentioned suggestions, further modifications to the code may still be required to ensure that the application effectively simulates the behavior of running two independent applications within a single instance, with each managing its own message types and priorities.

Parameter Settings:

The parameters used in this evaluation are listed in Table 1. Message generation within each vehicle follows a Poisson process. Make sure to use channel switching configuration.

Table 1: Parameter settings

		Parameter	Value
General		Simulation time	400 sec
		Road type	Ring road
		Road length	3040 m
		Number of lanes	2 lanes in the same direction
		Number of vehicles at any given time	{1, 2, 4, 6, 8, 16, 32}
		Vehicle average speed	13.89m/s
		Vehicle average acceleration	3.1 m/s^2
		Vehicle average deceleration	3.4m/s^2
Application	App1	Message length	250 Byte
		Average WSM rate	400 msg/sec
		Data user priority (UP1)	{0 ~ 7}
	App2	Message length	250 Byte
		Average WSM rate	400 msg/sec
		Data user priority (UP2)	{0 ~ 7}
MAC / PHY		Minimum contention window size (aCW_{min})	{5 ~ 1023} slot
		Maximum contention window size (aCW_{max})	1023 slot

	Queue size	unlimited
	Transmitted power	10mW
	Transmission rate	12Mbps
	Service channel	CH174
	Channel bandwidth	10Mhz
	Antenna type and placement	monopole antenna on roof
	Analog path loss model	SimplePathLossModel

Performance Metrics

1. Average packet delay: We define packet delay as the time interval from when a packet is generated at the application level of the sender to the time it is received by the application of the RSU.
2. Packet delivery ratio at application (aPDR): aPDR measures the reliability of the network at the application level by calculating the ratio of successfully delivered packets at the RSU to the total number of packets generated by all vehicles within the RSU coverage.

Tasks:

Perform the following tasks. The tasks are divided into two parts. Note that all measurements are in SCH, and you may need to use the repeat option in the ini file to get smooth graphs.

Part 1:

1.1. Modify the **ieeeV2X_2** project so that the RSU periodically broadcasts two WSA messages during each CCHI, advertising two services on the same Service Channel (SCH 174). When a vehicle receives a WSA, it should begin broadcasting the corresponding WSMs for the associated service, if it is not already doing so. **App1** should use **UP1**, while **App2** should use **UP2** for message prioritization. Try to avoid modifying files in the veins project. However, if you do modify any file in the veins project, you need to mention those modifications in your report. Package all modified files in the ieeeV2X_2 project (including your ini file) and submit them with your report.

1.2. Number of internal collisions: If UP1 is 5, and UP2 is 3, draw a box plot where the y-axis represents the number of internal collisions (internal contentions) for a vehicle, and the x-axis represents values for aCWmin (5 ~ 1023). For the values of aCWmin, choose step sizes so that the resulting graph is not too crowded. Let the number of vehicles at a given time be 8. The box plot should clearly display the average, minimum, and maximum number of internal collisions per vehicle for each measured value of aCWmin. Discuss the results.

1.3. Packet delivery ratio at application (aPDR): If UP1 is 5, and UP2 is 3, draw two graphs (one for App1 and the other for App2) with the y-axis representing the

application-level Packet Delivery Ratio (aPDR) for the corresponding App and the x-axis representing various values of the minimum contention window (aCWmin). The graph should include the results for different numbers of vehicles {1, 2, 4, 6, 8, 16, 32} plotted on the same graph. Discuss and compare both graphs.

1.4. Number of internal collisions: If UP1 is 5, and UP2 takes the following values {1, 2, 0 , 3, 4, 5, 6, 7} (in this order), draw a box plot with the y-axis representing the number of internal collisions (average, min, and max) for a vehicle and the x-axis representing various values of UP for App2. Let the number of vehicles be 8 and aCWmin be 15. Discuss your results. What happens when UP2 equals 4 and 5?

Part 2:

The SUMO traffic scenario has been modified. It now consists of two intersecting roads, each with two lanes in both directions, as shown in Figure 3. The intersection is controlled by all-way stop signs. The average vehicle speed on these roads is 13.89 meters per second. Whenever a vehicle exits the communication range of the RSU or reaches its destination, a new vehicle is randomly placed within the range, ensuring a constant number of vehicles at all times. The RSU is positioned at the intersection. Assume an ideal wireless channel, meaning no errors occur due to noise or other impairments. All other settings should remain consistent with those in Part 1. You may need to use **Netconvert** and **netedit** tools included with SUMO.

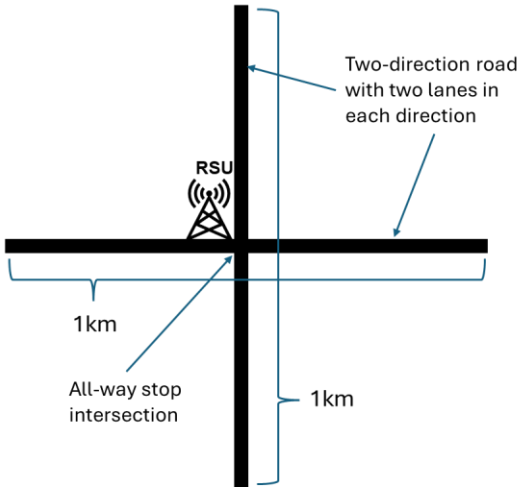

Figure 3: Physical layout of the new scenario

2.1. Modify the SUMO traffic scenario as explained. Package all new SUMO files in an archive file and submit it with your report.

2.2. Repeat tasks 1.2 to 1.4 using the new SUMO scenario, and compare the results.

Deliverables:

Each group is required to submit one report and the modified files in the ieeeV2X_2 project through LEARN. The report must be uploaded as a single PDF file to your group's designated Dropbox on the LEARN. Ensure that the report covers all required tasks and is formatted correctly. Package your modified ieeeV2X_2 project as an archive file, including any modified files from the Veins project. The archive file should also include all new SUMO-related files from part 2. Name your archive file as "LAB2_group{number}". For example, group 100 would name their archive file as "LAB2_group100".

Report contents:

- The report should contain answers to the questions outlined in the tasks. Provide detailed explanations and, where applicable, support your answers with diagrams or mathematical equations.
- For any plots or graphical data representations, we strongly recommend using MATLAB or Python, as these tools provide better precision and flexibility for the types of plots needed in this lab.

The report should be well-organized, clear, and concise. Include a cover page with the following information: Lab Number, Names, and student IDs.

Submission:

- "The report must be submitted by the deadline specified in the course outline. Ensure that you have uploaded the correct files before the deadline, as late submissions may be subject to penalties.
- If you encounter any issues with submission, contact the TA well before the deadline to resolve them.